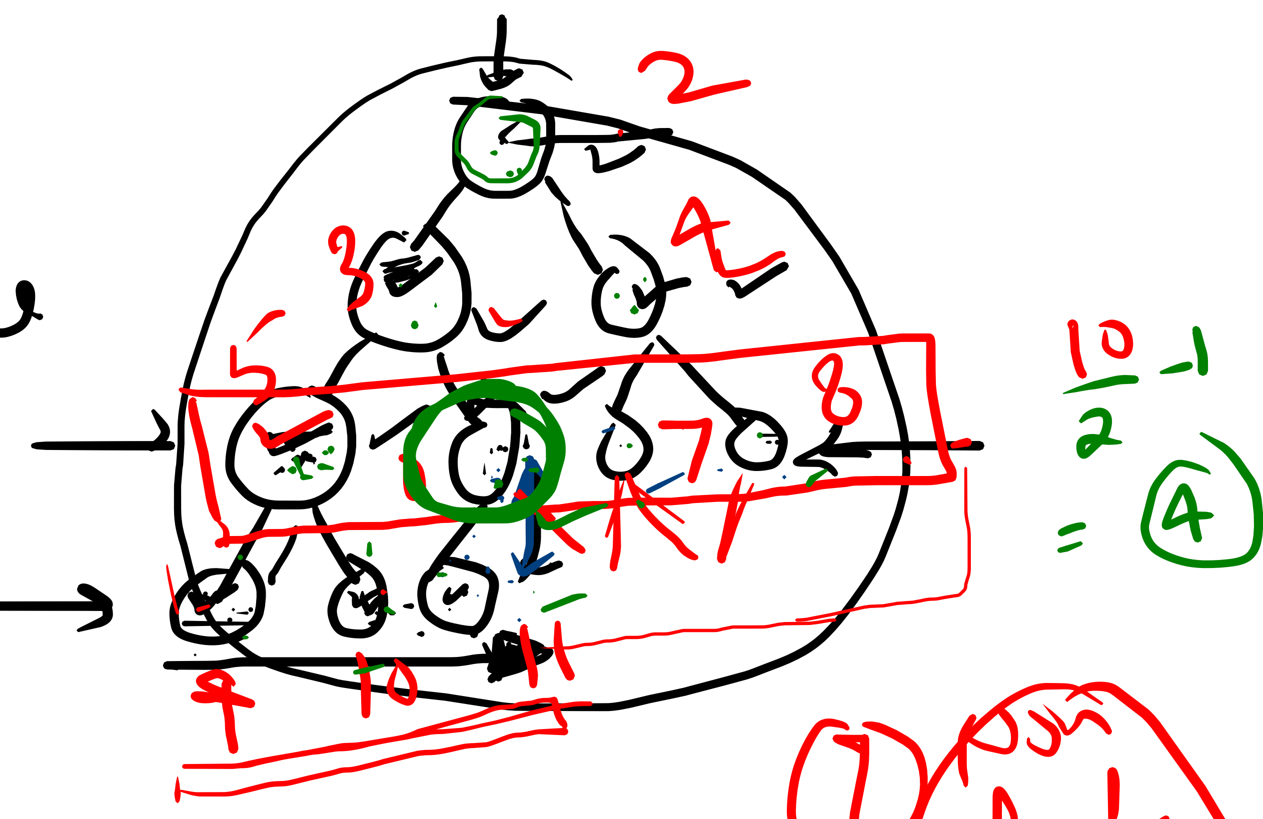
Heap :- Binary Heap - Tree structure

↳ Complete Binary Tree

↳ Max Heap → value at Node > Left value & Right value

↳ Min Heap → Node value < Left value & RV



Application → Heap sorting = O(n log n)

↳ Priority Queue — Operating System

Del values

→ Creation of Heap ✓

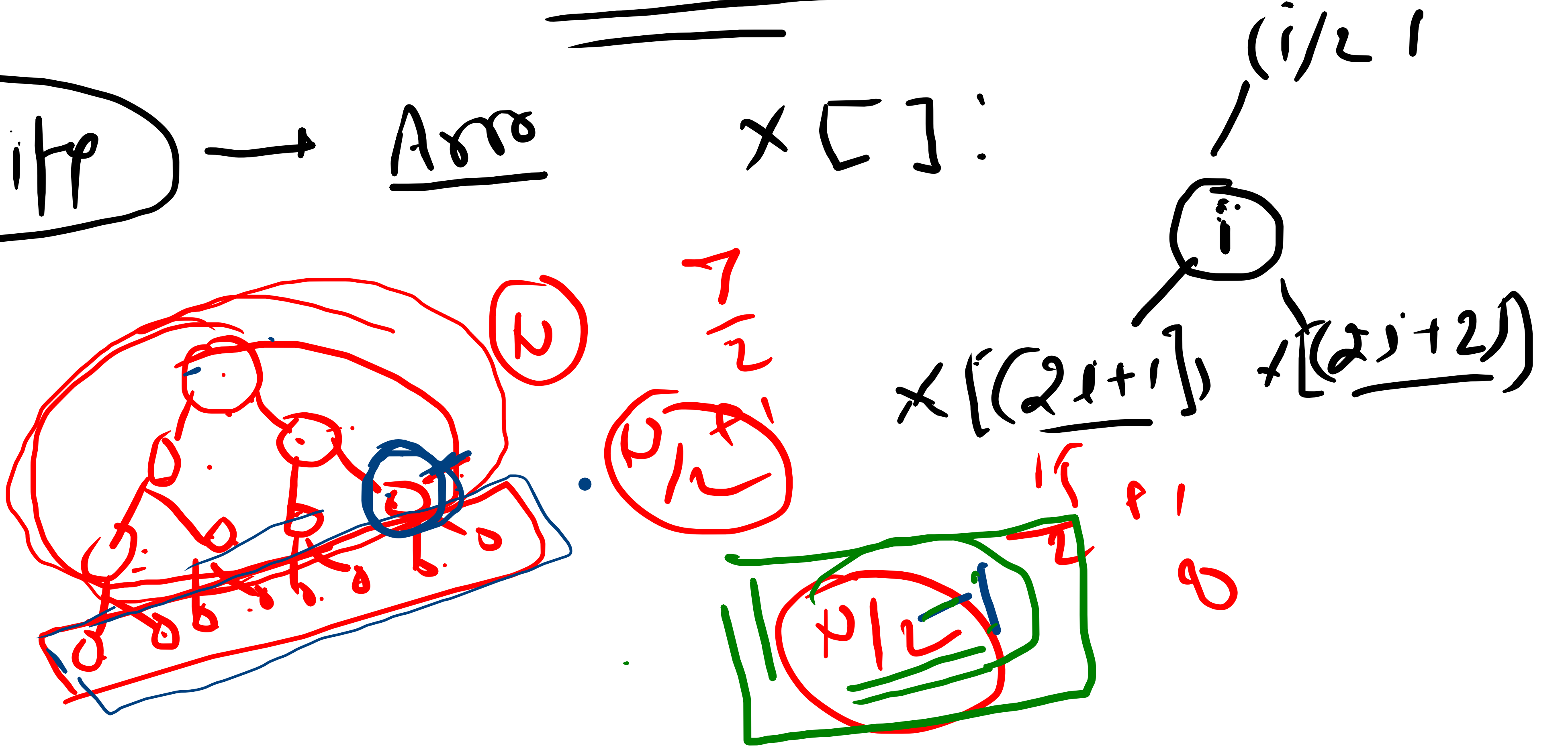
→ Insertion ✓

→ ~~Delt~~ Deletion ✓

→ Traverse ≠ Binary Tree

→ Heap Sort

Heapify → Arr x[ ]:



① Creation of Heap! → [Data is given in Array

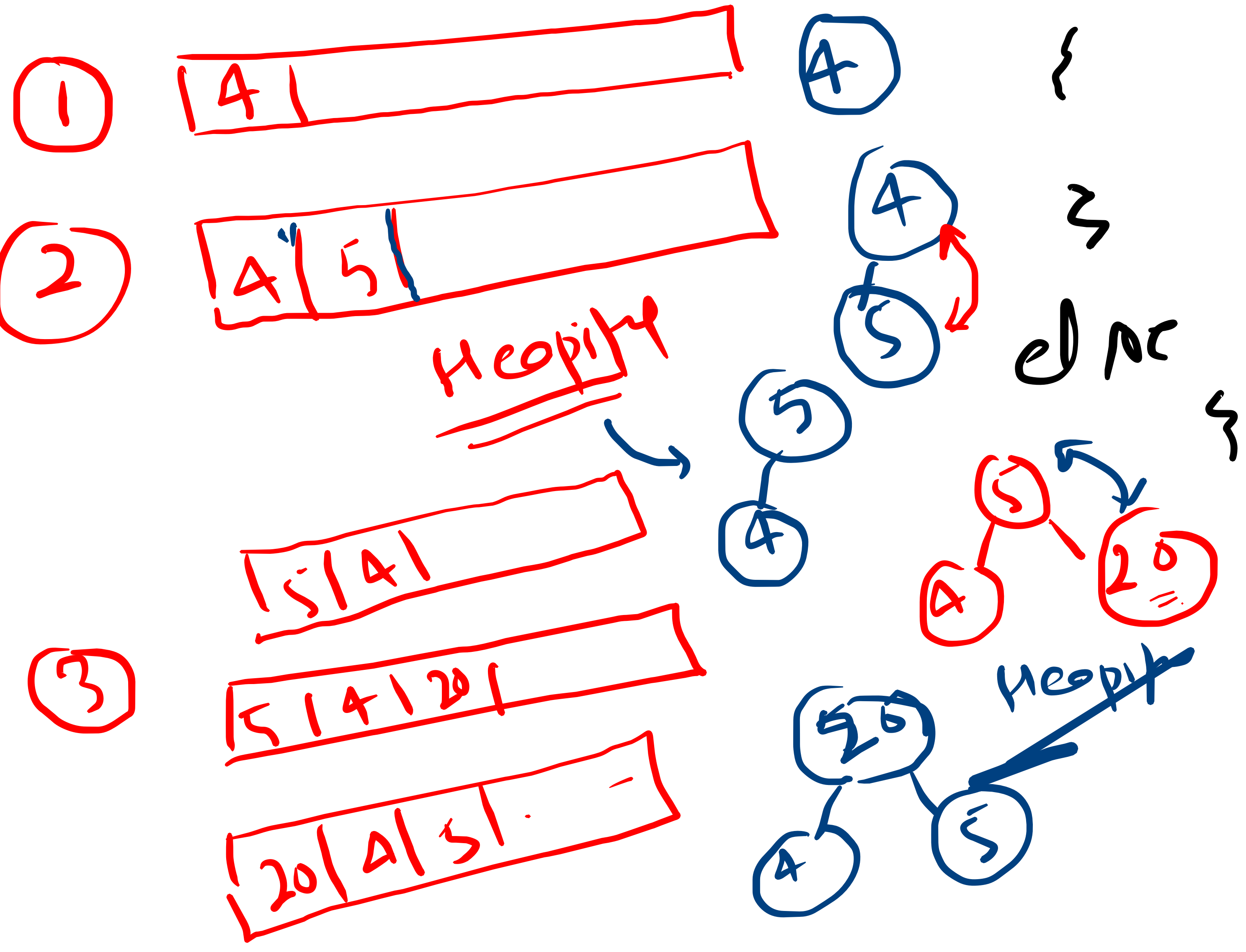
→ ARRAY implementation

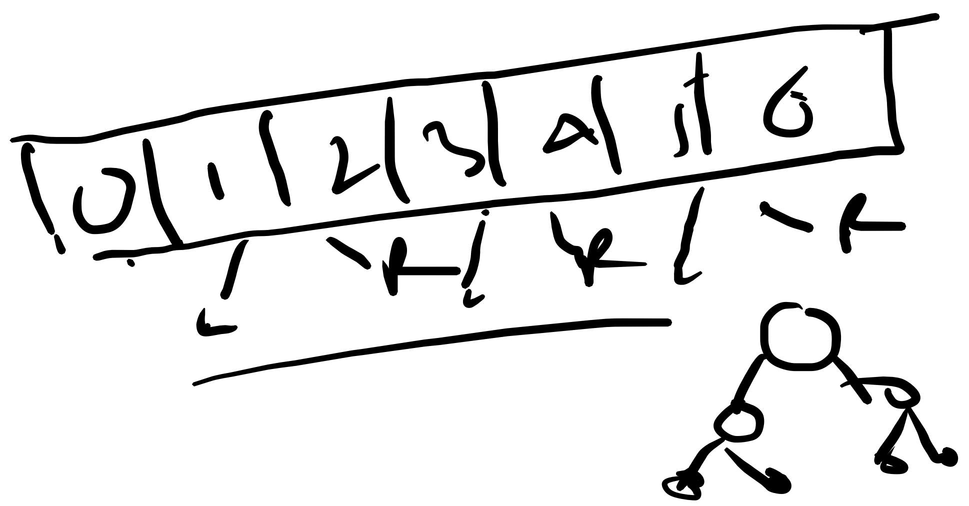
```
Void Heap Insert( int Array[ ] , int value)
{
    if (size == 0)
    {   Array [size] == value;
        size++;
    }
    else
    {
        Array [size]= value;
        size++;
        for( int i = ~~size/2-1~~ ; i>=0; i--)
            Heapify( Array , size , i);
    }
}
```

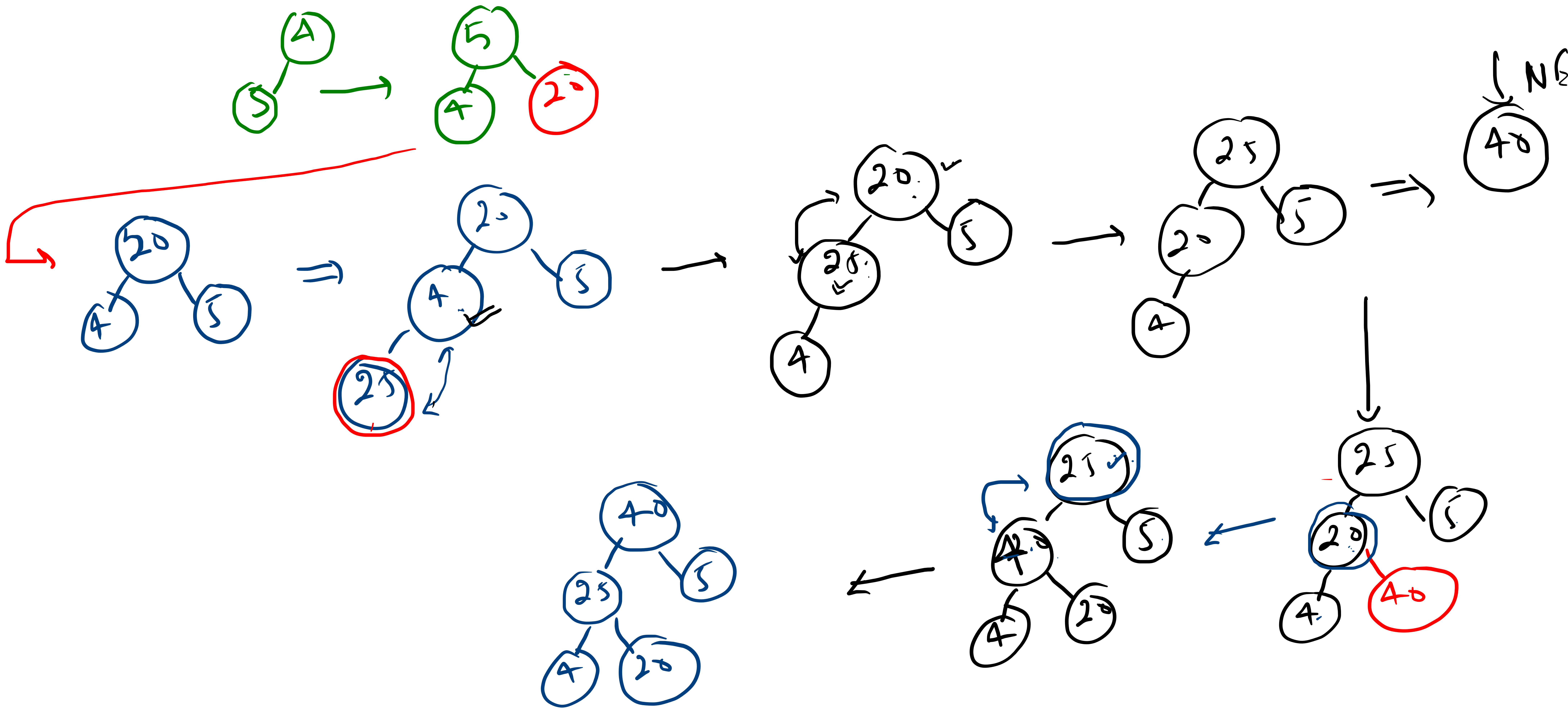
```
int size = 0;
int Main
{   int N=10;
    int X[N]; ←
    Heap insert( X , 4);
    Heap insert( X , 5);
    Heap insert( X , 10);
    ( X , 20);
}
```

Last leaf-Node

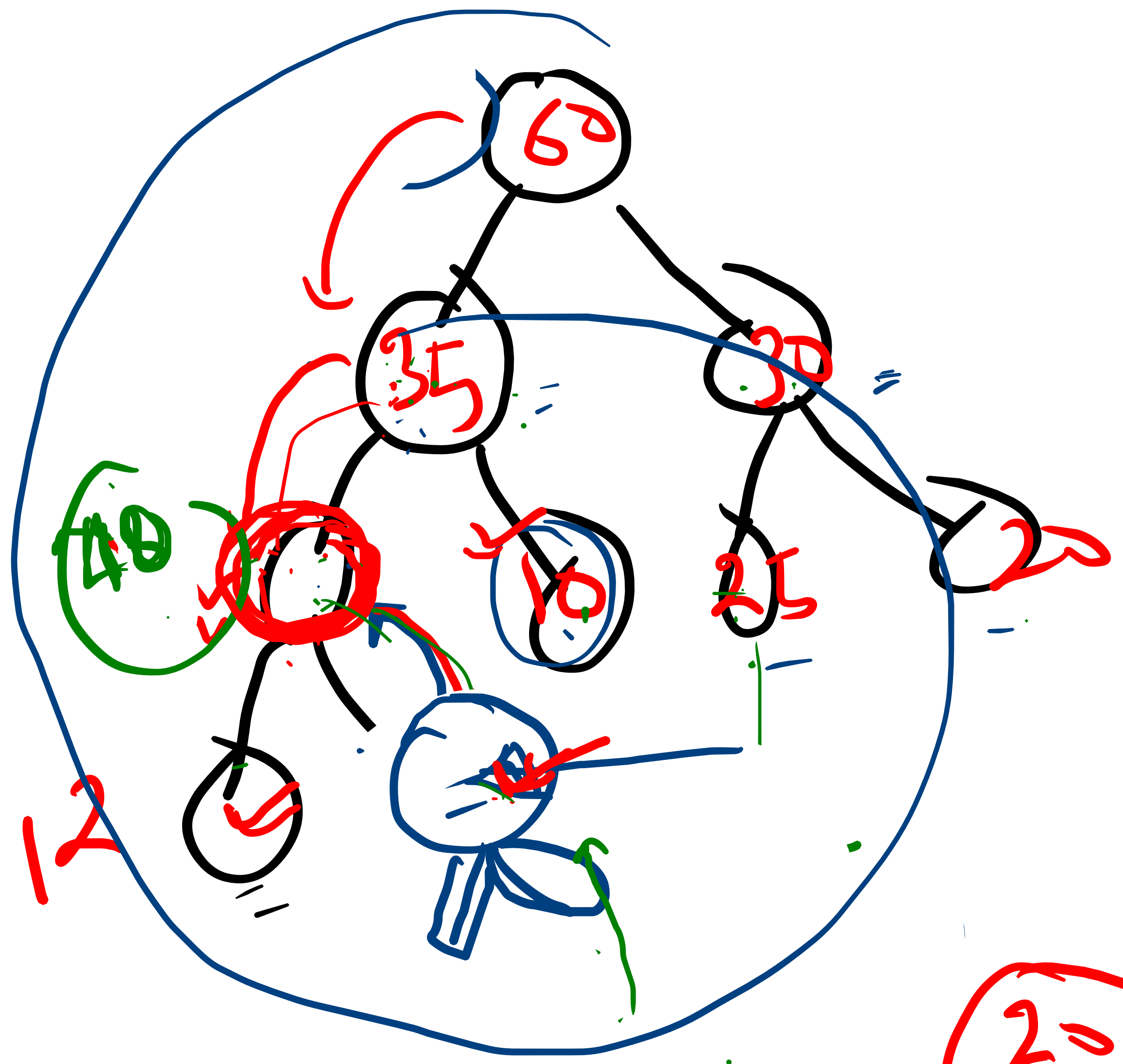
N/2-1

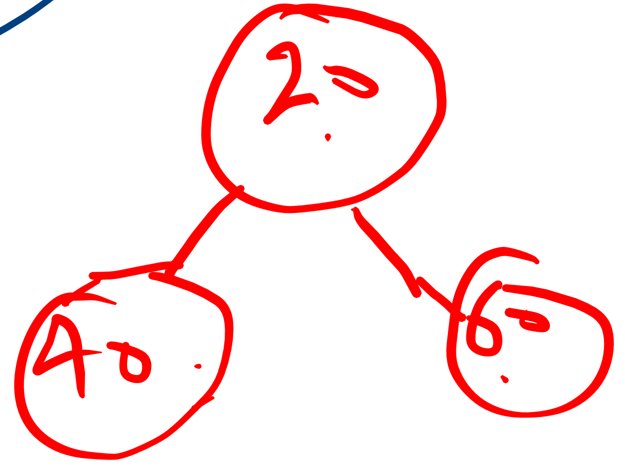


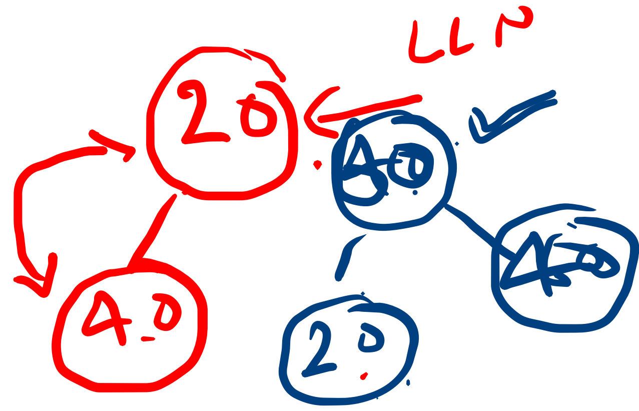




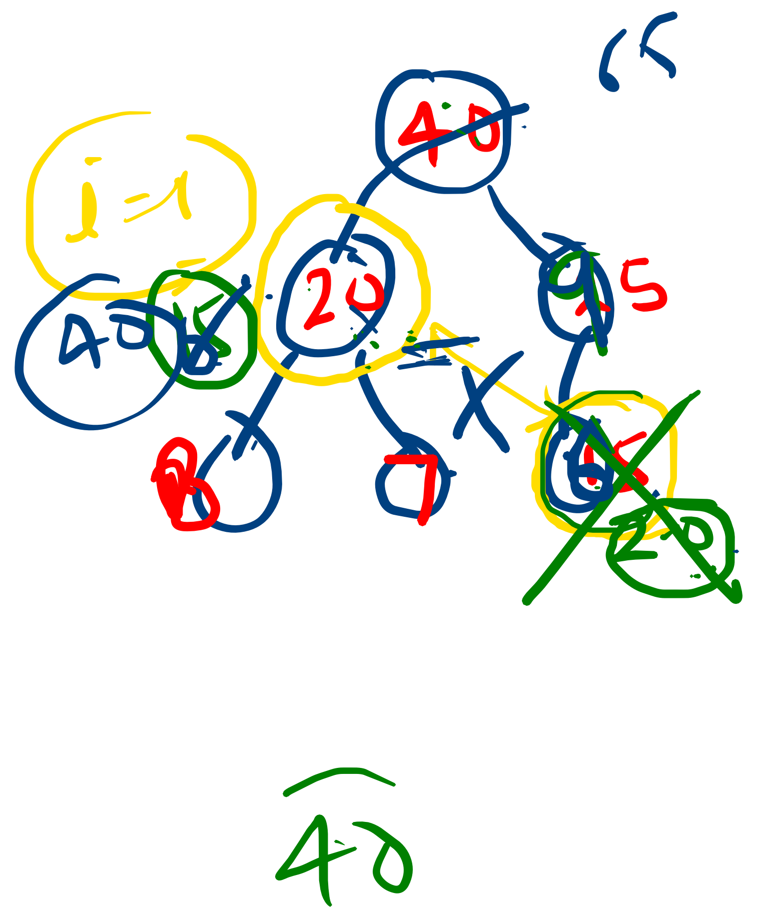
```
Heapify (int x[], int size, int index)
{
    if (size = = 1)
        return;
    else
    {   int Largest = index;
        int L = 2*index +1;
        int R = 2*index +2;
        if ( L < size && x[L] > x[Largest];
            Largest = L;
        if ( R < size && x[R] > x[Largest];
            Largest = R;
        if ( largest != index)
        { swap (x[index], x[Largest]);
          Heapify ( x, size, Largest);
    }
}
```



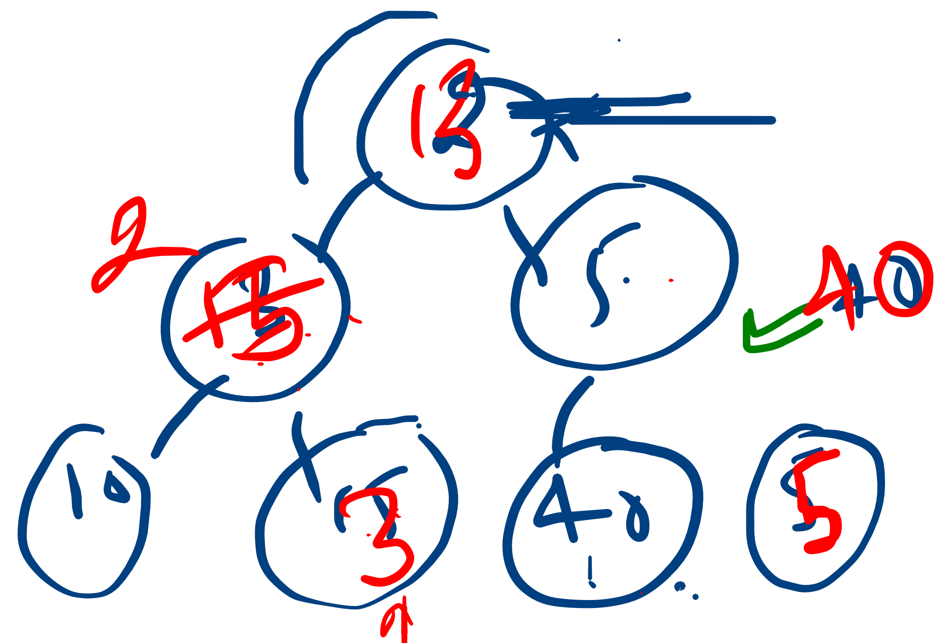


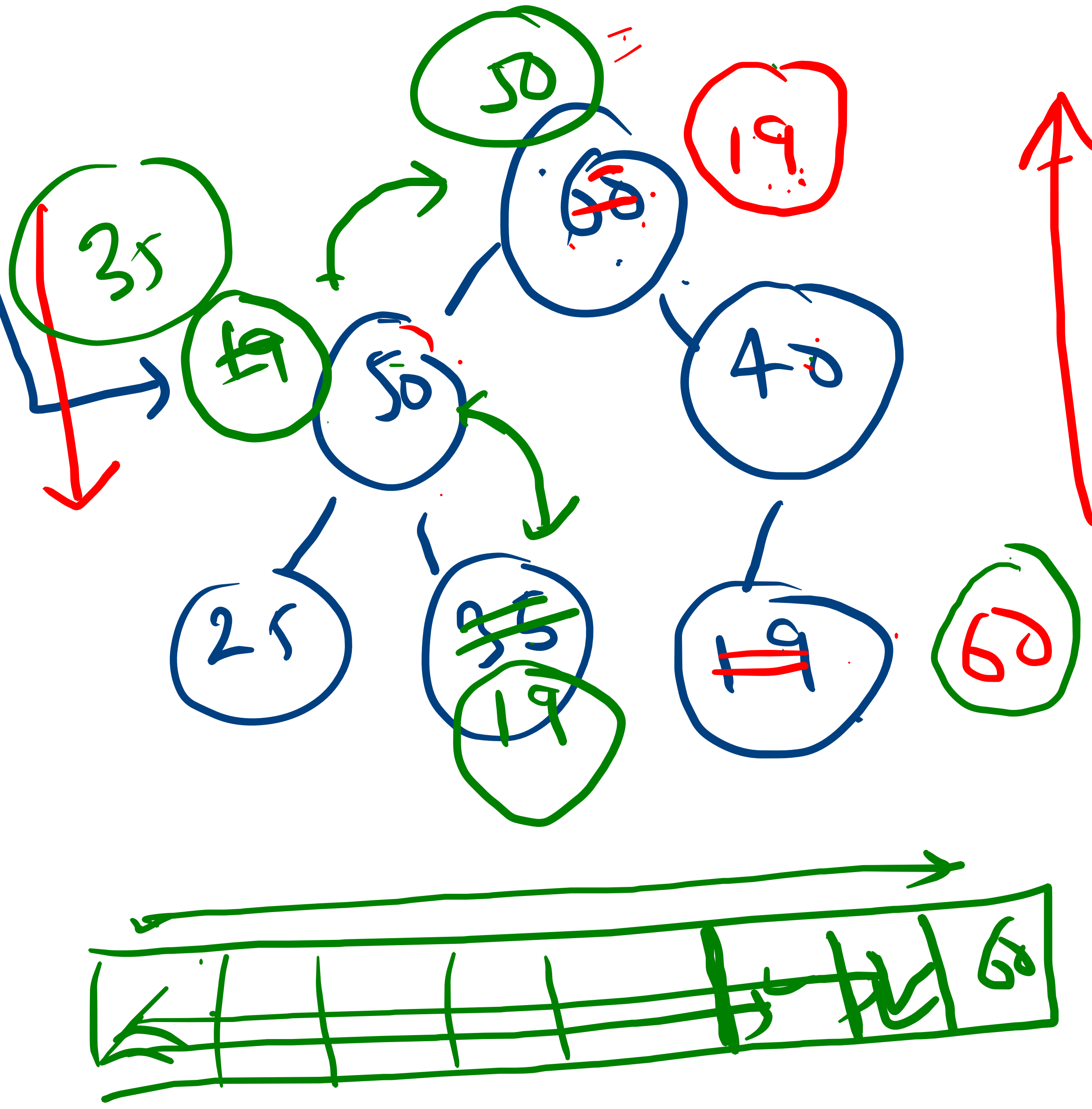


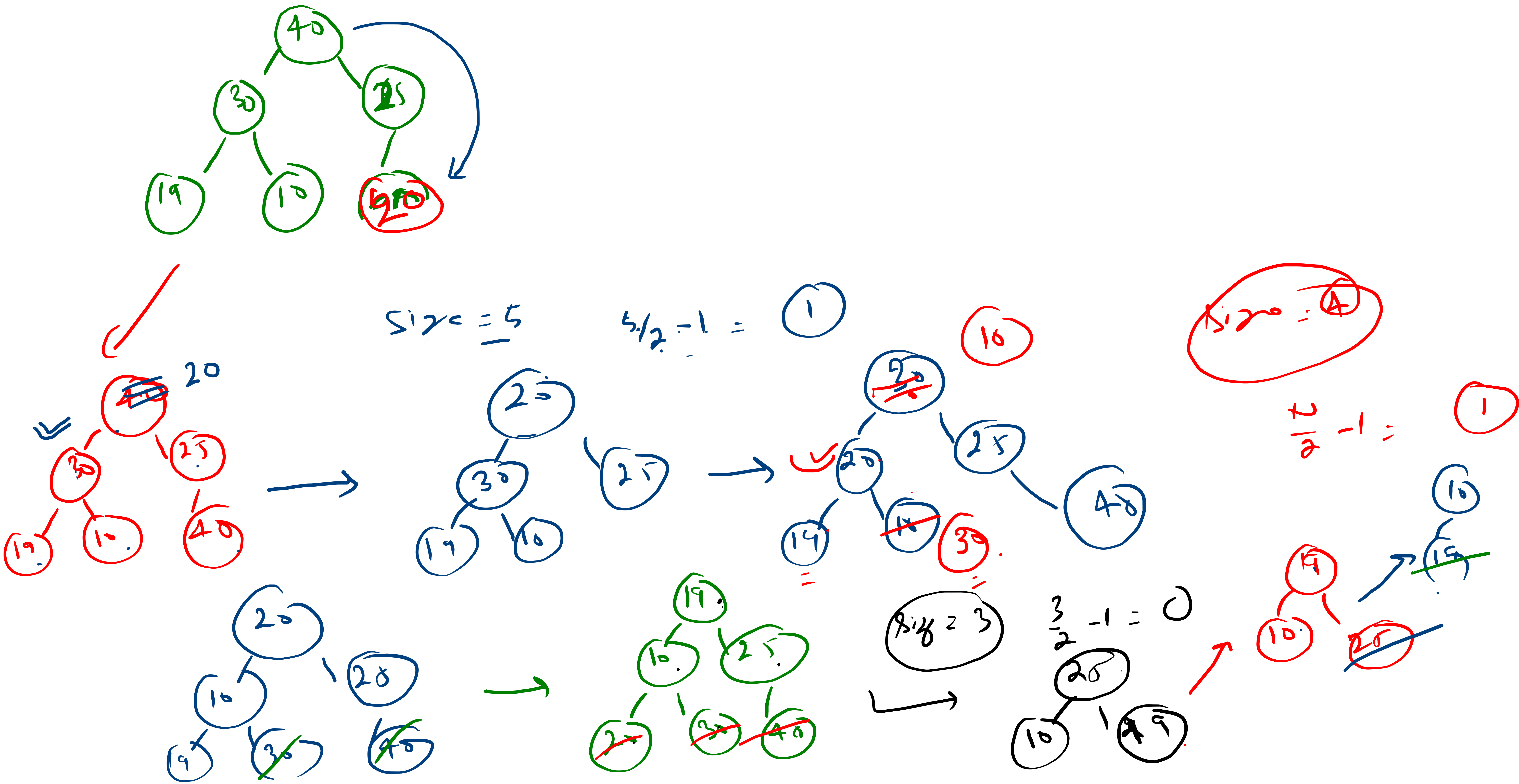
```
Void Delete ( int x[] , int value)
{
        int i = 0
        For ( i=0, i< size ; i++)
          {
             if ( value = x[i]
                 Break;
          }
     Swap ( x[i] , x[size-1]);
       size = size -1)
       For ( i = $\frac{size}{2}$ -1 ; i>=0; i--)
              Heapify ( x, size, i);

}
```



```
main ( )
{
    int x = { 2, 3, 5, 10, 15, 40 };
    int size = sizeof(x) / sizeof x[0];
    for (i = size/2 - 1, i >= 0; i--)

            Heapify ( x, size, i);

        Heap Sort ( x, size);
}

Heap Sort ( int x[], int size)
{
        for (i = size - 1 ; i > 0 ; i--)
            swap ( x[0], swap (i);
            heapify ( x, i, 0)

  3
```







size = 5
5/2 - 1 = 1
size = 4
4/2 - 1 = 1
size = 3
3/2 - 1 = 0

① Binary Tree $\longrightarrow$ Heap ??

```
Node* createHeap( Node* Root , int value);
{
        if(Root = = NULL)
        {
              Node* = createNode ( Value) ;
                return (Node);
        }

        else

}
```